GCU Glasgow Caledonian University

University for the Common Good

A vectorized bimodal distribution based micro differential evolution algorithm (VB-mDE)

Chen, Xu; Miao, Xueliang; Tianfield, Huaglory

Published in:
Multiagent and Grid Systems

DOI:
10.3233/MGS-200331

Publication date:
2020

Document Version
Author accepted manuscript

Link to publication in ResearchOnline

Citation for published version (Harvard):
Chen, X, Miao, X & Tianfield, H 2020, 'A vectorized bimodal distribution based micro differential evolution algorithm (VB-mDE)', *Multiagent and Grid Systems*, vol. 16, no. 3, pp. 245-261. https://doi.org/10.3233/MGS-200331

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please view our takedown policy at https://edshare.gcu.ac.uk/id/eprint/5179 for details of how to contact us.

Download date: 05. May. 2023

Glasgow Caledonian University

University for the Common Good

A vectorized bimodal distribution based micro differential evolution algorithm (VB-mDE)

Chen, Xu; Miao, Xueliang; Tianfield, Huaglory

Published in:
Multiagent and Grid Systems

Publication date:
2020

Document Version
Peer reviewed version

Link to publication in ResearchOnline

Citation for published version (Harvard):
Chen, X, Miao, X & Tianfield, H 2020, 'A vectorized bimodal distribution based micro differential evolution algorithm (VB-mDE)', *Multiagent and Grid Systems*, vol. 16, no. 3.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please view our takedown policy at https://edshare.gcu.ac.uk/id/eprint/5179 for details of how to contact us.

Download date: 24. Sep. 2020

A Vectorized Bimodal Distribution based Micro Differential Evolution Algorithm (VB-mDE)

Xu CHEN [a,1], Xueliang MIAO [b] and Hugo TIANFIELD [c]

[a] *School of Electrical and Information Engineering, Jiangsu University, Zhenjiang 212013, Jiangsu, China*

[b] *China Nuclear Power Technology Corporation, LTD, Wuhan, 430223, Hubei, China*

[c] *School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK*

Abstract. Micro differential evolution (mDE) refers to algorithms that evolve with a small population to search for good solutions. Although mDEs are very useful for resource-constrained optimization tasks, the research on mDEs is still limited. In this paper, we propose a new mDE, i.e., vectorized bimodal distribution based mDE (called VB-mDE). The main idea is to employ a vectorized bimodal distribution parameter adjustment mechanism in mDE for performance enhancement. Specifically, in the VB-mDE, two important control parameters, i.e., scale factor F and crossover rate CR, are adjusted by bimodal Cauchy distribution. At the same time, to increase the population diversity, the scale factor F is vectorized. The proposed VB-mDE is evaluated on the CEC2014 benchmark functions and compared with the state-of-the-art mDEs and normal DEs. The results show that the proposed VB-mDE has advantages in terms of solution accuracy and convergence speed.

Keywords. Micro differential evolution, small population, bimodal distribution, vectorized bimodal Cauchy distribution, parameter adjustment mechanism

1. Introduction

Differential evolution (DE) is one of the most powerful evolutionary algorithms for global optimization problems [1] . During the past two decades, DE has received much attention due to its attractive characteristics such as simplicity, speediness and robustness. DE has also been successfully applied to solve various scientific and engineering problems, such as chemical process optimization [2], economic load dispatch [3], and flow shop scheduling [4].

Normal DEs usually works on a large population size [5]. With a large population size, DE has better population diversity during the search process, and also has a higher opportunity to achieve global solutions for complex problems. However, for the resource-constrained problems like on-line nonlinear model predictive control (NMPC) [6] or real-

[1]Corresponding Author: School of Electrical and Information Engineering, Jiangsu University, Zhenjiang 212013, Jiangsu, China; E-mail:xuchen@ujs.edu.cn

time vehicle navigation system [7], the computing resources available for population re-evaluation in one iteration are largely restricted. In such circumstances, DEs with large population may become ineffective.

Recently, micro-DE (mDE) algorithms, also called as μDE, have been proposed as good alternatives. mDE utilizes a small population size, which is often set under 10. They have fast convergence speed, and can be implemented in the resource-constrained problems, e.g., online applications. However, mDE often suffers premature convergence and stagnation due to the lack of population diversity. To overcome the weakness, several modified mDE algorithms have been developed [8,9,10,11,12,13,14] .

Although mDE algorithms can be very useful on the special scenarios, the research on the mDE algorithms is still little compared to normal DE algorithms. On the other hand, it is recognized that the parameter adjustment mechanism (PAM) has important impacts on the performance of DE algorithms [15,16].

In this paper, we focus on advanced PAM for mDE algorithms. Specifically, using vectorized bimodal distribution based PAM (VB-PAM), we propose a novel vectorized bimodal distribution based micro-DE (VB-mDE). In VB-mDE, two important control parameters, i.e., scale factor F and crossover rate CR, are adjusted by bimodal Cauchy distribution. At the same time, to increase the population diversity, the bimodal distribution of scale factor F is vectorized.

The bimodal distribution based PAM is taken from the normal DE algorithm in the literature [17]. In this paper, we will show how the bimodal distribution based PAM can be used to design novel mDE algorithm. Our proposed VB-mDE will be compared with several state-of-the-art mDE and normal DE algorithms on the CEC2014 functions under the scenarios of small computational resources available for population re-evaluation.

The remainder of this paper is organized as follows. Section 2 reviews the existing research of mDE algorithms. Section 3 proposes our VB-mDE algorithm. Section 4 evaluates our proposed VB-mDE algorithm with comprehensive simulation results and analysis. Section 5 draws conclusion and gives an outlook on future work.

2. Literature review

This section will briefly review the research work of the mDE algorithms. The main methods of mDE include (i) modification of search operators, (ii) local search based method, and (iii) adjustment of parameters or operators, among others.

The first method to develop efficient mDE algorithms may be modifying the search operators. For example, to address the prematurity problem, a modified DE using smaller population called DESP is developed in [9]. In DESP, the disturbance is introduced to the mutation operator, and an adaptive scheme is also used to adjust the disturbance size. For the purpose of enhancing the population diversity, the random perturbation and modified selection strategies are introduced to the mDE [10]. It is shown that the two modifications can significantly improve the performance of mDE. However, the mDE algorithms developed in [9,10] use constant control parameters and were only evaluated in simple test functions. Therefore, it is difficult for them to obtain good performance in complex optimization functions.

The second method may be employing local search technique to design efficient mDE algorithms. In [11], a mDE algorithm with local search operator, called mDELS,

is developed and applied for solving large-scale optimization problems. In [12], a mDE algorithm with a directional local search (called μDSDE) is proposed to solve large-scale problems. In μDSDE, exploration is realized by reinitializing the worse individuals, while exploitation is performed through mutation, crossover and directional local search. In [11,12], the local search technique increases the exploitation ability of the mDE algorithms. Meanwhile, it also suffers the risk of local convergence.

The third method may be devising parameter and operator adaptive strategies for mDE algorithms. In [13], a mDE algorithm with vectorized random mutation factor called MDEVM is proposed. In MDEVM, by randomizing and vectorizing the scale factor, the diversity of the population can be increased, thereby alleviating the problems of premature and stagnation. Later, by employing ensemble mutation and oppositional learning strategies in MDEVM, the ensemble mDE (EMDE) [14] and oppositional ensemble mDE (OEMDE) [18] are presented, resectively. In the mDE algorithms [13,14,18], the parameter or operator adjustment strategies are devised based on random uniform distribution, and thus the performance is limited. In [6], an adaptive mDE algorithm called μJADE is proposed. μJADE is developed based on the JADE [19]. In μJADE, performance is achieved through implementing a combination of parameter adaptive strategy together with 'current-by-rand-to-pbest' mutation, perturbation strategy and restart strategy. However, the μJADE algorithm has quite high complexity of implementation.

Furthermore, some mDE algorithms are designed for real-world optimization problems. To deal with the image thresholding problem, opposition-based population initialization (OBPI) is embedded into mDE, and a mODE algorithm is proposed in [8]. It is shown that mODE converges faster than mDE through the OBPI. To solve the optimization problems of topological active net, mDE with best improvement local search (deBILS) is proposed in [20].

Compact DE (cDE) [21] is a related work to mDE. In cDE, a statistical description of the population is used to evolve the search process and the memory requirement is a population of four individuals.

The existing mDE algorithms may be summarized in Table 1.

Table 1. Summary of existing mDE algorithms

mDE method	mDE Algorithm	Main characteristics
Modification of search operators	DESP [9]	Adaptive disturbance based mutation operator
	mDE [10]	Random perturbation; modified selection strategies
Local search based method	mDELS [11]	Local search
	μDSDE [12]	Directional local search; reinitialize the worse individuals
	deBILS [20]	Best improvement local search
Adjustment of parameters or operators	MDEVM [13]	Vectorized random scale factor
	EMDE [14]	Vectorized random scale factor; ensemble five mutation operators
	OEMDE [18]	Vectorized random scale factor; ensemble five mutation operators; opposition-based learning
	μJADE [6]	Advanced adaption of scale factor and crossover rate; 'current-by-rand-to-pbest' mutation; perturbation strategy; restart strategy
Applications of mDE	mODE [8]	Image thresholding problem
	deBILS [20]	Topological active net optimization problems
Virtual population	Compact DE [21]	Virtual population with statistical representation

3. Proposed VB-mDE

This section proposes a new mDE algorithm, i.e., vectorized bimodal distribution based micro-DE algorithm (called VB-mDE). Our proposed VB-mDE algorithm can be illustrated by block diagram as in Figure 1.

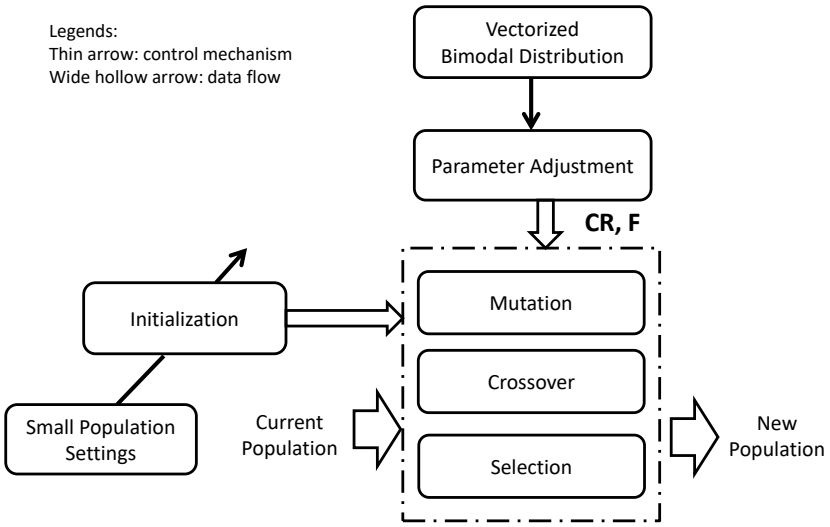


Figure 1. Block diagram of proposed VB-mDE algorithm

3.1. Basics of DE

DE randomly initializes a population of Np individuals in the search space. It is assumed that the dimension of the problem is D, the i-th individual in the population is:

$$\boldsymbol{X}_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,j}, \cdots x_{i,D}\} \tag{1}$$

where $x_{i,j}$ is the j-th dimension of the i-th individual. Initialization of the population is carried out as follows:

$$x_{i,j} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \times rand \tag{2}$$

where $x_j^{\max}$ and $x_j^{\min}$ denote the upper and lower bounds of the j-th dimension across the population, respectively, and *rand* is a random number within [0,1].

DE works over three phases in order, namely initialization, search and termination. After initialization, DE implements three operators, namely mutation, crossover and selection to generate the offspring of the population in the search, until the termination condition is satisfied.

The mutation operator is firstly applied to generate new offspring. The most commonly used mutation strategies include:

DE/rand/1:

$$\boldsymbol{V}_i = \boldsymbol{X}_{r1} + F(\boldsymbol{X}_{r2} - \boldsymbol{X}_{r3}) \tag{3}$$

DE/rand/2:

$$\boldsymbol{V}_i = \boldsymbol{X}_{r1} + F(\boldsymbol{X}_{r2} - \boldsymbol{X}_{r3}) + F(\boldsymbol{X}_{r4} - \boldsymbol{X}_{r5}) \tag{4}$$

DE/best/1:

$$\boldsymbol{V}_i = \boldsymbol{X}_{best} + F(\boldsymbol{X}_{r1} - \boldsymbol{X}_{r2}) \tag{5}$$

DE/best/2:

$$\boldsymbol{V}_i = \boldsymbol{X}_{best} + F(\boldsymbol{X}_{r1} - \boldsymbol{X}_{r2}) + F(\boldsymbol{X}_{r3} - \boldsymbol{X}_{r4}) \tag{6}$$

DE/current-to-best/1:

$$\boldsymbol{V}_i = \boldsymbol{X}_i + F(\boldsymbol{X}_{best} - \boldsymbol{X}_i) + F(\boldsymbol{X}_{r1} - \boldsymbol{X}_{r2}) \tag{7}$$

where $\boldsymbol{X}_{r1}$, $\boldsymbol{X}_{r2}$, $\boldsymbol{X}_{r3}$, $\boldsymbol{X}_{r4}$ and $\boldsymbol{X}_{r5}$ are individuals randomly selected from the population, satisfying $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$, $\boldsymbol{X}_{best}$ is the best individual in the population, $\boldsymbol{V}_i$ is the mutation vector, also called donor vector, and F is the scale factor.

After mutation, the crossover operator is applied between $\boldsymbol{V}_i$ and $\boldsymbol{X}_i$ to generate the trial vector $\boldsymbol{U}_i$, as described below:

$$U_{i,j} = \begin{cases} V_{i,j} & if \ \ rand < CR \ \text{ or } \ j = j_{rand} \\ X_{i,j} & otherwise \end{cases} \tag{8}$$

where $U_{i,j}$ is the j-th dimension of the i-th trial vector $\boldsymbol{U}_i$, CR is the crossover rate, and j_{rand} is a random integer within $[1, D]$, which ensures that at least one element of $\boldsymbol{U}_i$ is from $\boldsymbol{V}_i$.

After crossover, the selection operator is applied between the target vector $\boldsymbol{X}_i$ and the trial vector $\boldsymbol{U}_i$, as shown below:

$$\boldsymbol{X}_i = \begin{cases} \boldsymbol{U}_i & if\ f(U_i) \leq f(X_i) \\ \boldsymbol{X}_i & otherwise \end{cases} \tag{9}$$

where $f(\boldsymbol{U}_i)$ and $f(\boldsymbol{X}_i)$ are the fitness values of $\boldsymbol{U}_i$ and $\boldsymbol{X}_i$, respectively; and $f(\boldsymbol{U}_i) \leq f(\boldsymbol{X}_i)$ means $\boldsymbol{U}_i$ is not worse than $\boldsymbol{X}_i$.

3.2. Vectorized bimodal distribution mechanism for mDE

The bimodal distribution PAM (B-PAM) was proposed in [17], aiming at efficiently combining the global exploration and local exploitation during the searching process. B-PAM has two advantages, i.e., (1) it can be coded very easily in DE; and (2) it is shown that B-PAM obtains consistently high performance [16,17].

In this paper, we will employ B-PAM for the mDE, and integrate it into our VB-mDE algorithm. Furthermore, to enhance the population diversity and reduce the risk of premature convergence, the vectorized mutation strategy [13] is introduced into B-PAM, resulting in a vectorized version of B-PAM, called VB-PAM. To be specific, the bimodal distribution based scale factor in VB-PAM is set as follows:

$$\boldsymbol{F}_i = [F_{\mathrm{i},1}, F_{i,2}, \cdots, F_{i,j}, \cdots, F_{i,D}] \quad (i = 1,2,\cdots,Np) \tag{10}$$

$$F_{i,j} = \begin{cases} randc_i(0.65, 0.1) & if\ rand < 0.5 \\ randc_i(1.5, 0.1) & otherwise \end{cases} \quad (j = 1,2,\cdots,D) \tag{11}$$

From Eq.(10), it can be seen that the scale factor $\boldsymbol{F}_i$ for the i-th individual is vectorized. In other words, different values of $F_{i,j}, j = 1,\cdots,D$ are used in different dimensions and different individuals [22].

In Eq.(11), $randc_i(\theta, \partial)$ is a random number obeying Cauchy distribution. The range of scale factor F is [0.1,1.5] [2], the same as in [13]. Moreover, if the value generated by $randc_i(0.65, 0.1)$ is smaller than 0.1, then it will be truncated to 0.1; and if the value generated by $randc_i(0.65, 0.1)$ is larger than 1.0, then it will be truncated to 1.0. Likewise, if the value generated by $randc_i(1.5, 0.1)$ is smaller than 1.0, then it will be truncated to 1.0; and if the value generated by $randc_i(1.5, 0.1)$ is larger than 1.5, then it will be truncated to 1.5.

The bimodal distribution based crossover rate is set as follows:

$$CR_i = \begin{cases} randc_i(0.1, 0.1) & if\ rand < 0.5 \\ randc_i(0.95, 0.1) & otherwise \end{cases} \tag{12}$$

In Eq.(12), the range of cross rate CR is [0,1], which is the same as in [17]. If the value generated by $randc_i(0.1, 0.1)$ is smaller than 0, then it will be truncated to 0; and if the value generated by $randc_i(0.95, 0.1)$ is larger than 1.0, then it will be truncated to 1.0.

In DE, the scale factor F controls the search range of mutation operator. A large value of F is helpful for global exploration, while a small value of F is beneficial for local

[2] In mDE, it is very important to keep the population diversity, and therefore the upper bound for scale factor F is set to be a fairly larger value (i.e., 1.5), which is recommended in [13].

exploitation. In Eq.(11), the vectorized scale factor F has 50 percent probability being within $[0.1, 1.0]^D$, which focuses on local search; and another 50 percent probability being within $[1.0, 1.5]^D$, which favors the global exploration.

In addition, a large value of CR is helpful for the diversity enhancement, while a small value of CR is helpful for the fast convergence. In Eq.(12), the crossover rate CR_i has 50 percent probability being distributed around 0.1, which puts emphasis on accelerating convergence, and the other 50 percent probability being distributed around 0.95, which puts emphasis on enhancing the population diversity.

3.3. Pseudocodes of VB-mDE

By using the VB-PAM and small population, our proposed VB-mDE algorithm can be represented by pseudocodes as shown in Algorithm 1.

The structure of the proposed VB-mDE algorithm would remain unchanged if the mutation strategy rand/1 is substituted with any other mutation strategies, such as rand/2, best/1, best/2 and current-to-best/1, in which case the VB-mDE algorithm accordingly turns to be a VB-mDE/a/b algorithm. In general, VB-mDE has a very simple structure, just like basic DE, and can be coded and implemented very easily.

3.4. Difference between VB-mDE and MDEVM

MDEVM [13] is a recently-developed mDE algorithm, which originally uses the vectorized mutation strategy. Since VB-mDE and MDEVM appear in great similarities, it is worthwhile to highlight the differences between the two algorithms.

The main difference between VB-mDE and MDEVM lies in the parameter adjustment mechanisms. Firstly, MDEVM generates the scale factor F based on a uniform distribution, while VB-mDE utilizes bimodal Cauchy distribution to generate the scale factor F. Secondly, MDEVM does not self-adjust the crossover rate CR but only uses a constant value $CR = 0.9$. By contrast, VB-mDE employs bimodal Cauchy distribution to generate the crossover rate CR.

Algorithm 1 Vectorized Bimodal Distribution based Micro-DE algorithm (VB-mDE)

g=0 // Initialization;
Input the small population size Np;
for i=1 to Np **do**
  for d=1 to D **do**
    $x_{i,d} = x_i^{min} + rand \times (x_i^{max} - x_i^{min})$
  end for
end for
$\boldsymbol{P}^g = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{Np}\}$
while the termination condition is not met **do**
  for i=1 to Np **do**
    //Mutation, exemplified by mutation strategy rand/1
    $\boldsymbol{F}_i = [F_{i,1}, F_{i,2}, \cdots, F_{i,d}, \cdots, F_{i,D}]$ is generated by bimodal distribution based on Eq. (11)
    Select three individuals from population $\boldsymbol{P}^g$ satisfying $\boldsymbol{X}_{r1} \neq \boldsymbol{X}_{r2} \neq \boldsymbol{X}_{r3} \neq \boldsymbol{X}_i$
    for d=1 to D **do**
      $V_{i,d} = X_{r1,d} + F_{i,d}(X_{r2,d} - X_{r3,d})$
    end for
    //Crossover
    CR_i is generated by bimodal distribution based on Eq. (12)
    if $rand < CR_i$ or $j = j_{rand}$ **then**
      $U_{i,d} = V_{i,d}$
    else
      $U_{i,d} = X_{i,d}$
    end if
    //Selection
    if $f(\boldsymbol{U}_i) \leq f(\boldsymbol{X}_i)$ **then**
      $\boldsymbol{X}_i^{new} = \boldsymbol{U}_i$
    else
      $\boldsymbol{X}_i^{new} = \boldsymbol{X}_i$
    end if
  end for
  $\boldsymbol{X}_i = \boldsymbol{X}_i^{new}, \quad \forall i \in \{1, 2, \cdots, Np\}$
  g=g+1 // Update generation counter
  $\boldsymbol{P}^g = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{Np}\}$ // Update population
end while

4. Performance evaluations

The proposed VB-mDE is compared with the state-of-the-art mDE and nomal DE algorithms to validate its performance. All the algorithms are tested on 30 benchmark functions from CEC2014 with four different dimensions, i.e., D=10, 30, 50 and 100 [23]. The benchmark functions fall into four groups, i.e., (1) unimodal functions (F1∼F3); (2) simple multimodal functions (F4∼F16); (3) hybrid functions (F17∼F22); and (4) composition function (F23∼F30).

The comparisons are summarized using B-S-W triplets, in which B, S and W represent the numbers of test functions, respectively, on which VB-mDE performs better than, similarly to and worse than its competitor according to the Wilcoxon rank sum test at significant level $\alpha = 0.05$. The maximum number of functional evaluations is set to

$maxFES$=2000 $\times$ D [3], and all the algorithms are implemented 51 times independently. The simulations are conducted in MATLAB2014a and in a PC with Windows 7, Dual-core 3.9GHz CPU, 4GB RAM.

4.1. Comparisons between VB-mDE and MDEVM with different small population sizes

Firstly, we compare VB-mDE and MDEVM to investigate whether the vectorized bimodal distribution based PAM outperforms the vectorized uniform distribution based PAM in the mDEs.

We compare the performances of VB-mDE and MDEVM with different small population sizes. Table 2 shows the comparisons between VB-mDE and MDEVM with small population size $Np \in \{4,5,6,7,8,9,10\}$.

When $Np = 4$, VB-mDE outperforms MDEVM on 30, 30 and 24 functions, but loses merely to MDEVM on none, none and 2 functions under the mutation strategies rand/1, best/1 and current-to-best/1, respectively.

When $Np = 5$, VB-mDE outperforms MDEVM on 27, 30, 21 and 16 functions, but loses merely to MDEVM on none, none, 1 and none functions under the mutation strategies rand/1, best/1, best/2 and current-to-best/1, respectively.

When $Np = 6$, VB-mDE outperforms MDEVM on 24, 21, 23, 20 and 17 functions, but loses merely to MDEVM on 1, 8, none, 2 and 1 functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

When $Np = 7,8,9$ and 10, VB-mDE also outperforms MDEVM on most of the test functions under the five mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

Overall, VB-mDE outperforms MDEVM on 161, 116, 158, 125 and 123 functions, but loses merely to MDEVM on 10, 19, 10, 23 and 11 functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

From the comparisons it is clear that VB-mDE performs significantly better than MDEVM on most of the CEC2014 functions with different small population sizes. This validates the effectiveness of the vectorized bimodal distribution based PAM being built in VB-mDE.

[3]The recommended $maxFES$ value for the normal DE algorithms is $10000 \times D$ in [23]. In this study, the mDE algorithms use a reduced value $maxFES = 2000 \times D$.

Table 2. Comparisons between VB-mDE and MDEVM with small population sizes $Np \in \{4,5,6,7,8,9,10\}$ for dimension $D = 30$

	B-S-W: Mutation / Pop. size	rand/1	rand/2	best/1	best/2	current-to-best/1
	$Np = 4$	30-0-0	-	30-0-0	-	24-4-2
	$Np = 5$	27-3-0	-	30-0-0	21-8-1	16-14-0
VB-mDE	$Np = 6$	24-5-1	21-1-8	23-7-0	20-8-2	17-12-1
vs.	$Np = 7$	24-5-1	19-4-7	19-9-2	21-5-4	18-12-0
MDEVM	$Np = 8$	21-7-2	22-6-2	18-8-4	22-4-4	17-12-1
	$Np = 9$	18-9-3	26-2-2	19-8-3	21-3-6	16-11-3
	$Np = 10$	17-10-3	28-2-0	19-10-1	20-4-6	15-11-4
	Total B-S-W	161-39-10	116-15-19	158-42-10	125-32-23	123-76-11

B, S and W represent the numbers of test functions on which VB-mDE performs better than, similarly to, or worse than MDEVM, respectively.

4.2. Comparisons between VB-mDE and MDEVM with different problem dimensions

We further compare the performances of VB-mDE and MDEVM with four different problem dimensions. In the comparisons below, the population size for all mDE algorithms is set as $Np = 8$.

Table 3 shows the comparisons between VB-mDE and MDEVM with four different dimensions $D = 10$, 30, 50 and 100.

When $D = 10$, VB-mDE outperforms MDEVM on 21, 25, 27, 24 and 22 functions, but loses to MDEVM only on 3, 1, 2, none and none functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

When $D = 30$, VB-mDE outperforms MDEVM on 21, 22, 18, 22 and 17 functions, but loses to MDEVM only on 2, 2, 4, 4 and 1 functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

When $D = 50$ and 100, VB-mDE also outperforms MDEVM on most of the functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

When considering all four problem dimensions ($D = 10$, 30, 50 and 100), VB-mDE outperforms MDEVM on 84, 93, 85, 87 and 69 functions, but loses to MDEVM only on 12, 8, 10, 16 and 9 functions under the mutation strategies rand/1, rand/2, best/1, best/2 and current-to-best/1, respectively.

Based on the above analysis, it can be said that VB-mDE largely outperforms MDEVM under any problem dimension and with any mutation strategy. These comparisons further validate the effectiveness of the vectorized bimodal distribution based PAM being built in VB-mDE.

Table 3. Comparisons between VB-mDE and MDEVM with different problem dimensions

	B-S-W / Mutation / Dimension	rand/1	rand/2	best/1	best/2	current-to-best/1
	$D = 10$	21-6-3	25-4-1	27-1-2	24-6-0	22-8-0
VB-mDE	$D = 30$	21-7-2	22-6-2	18-8-4	22-4-4	17-12-1
vs.	$D = 50$	21-6-3	24-5-1	16-11-3	22-2-6	15-11-4
MDEVM	$D = 100$	21-5-4	22-4-4	24-5-1	19-5-6	15-11-4
	Total B-S-W	84-24-12	93-19-8	85-25-10	87-17-16	69-42-9

B, S and W represent the numbers of test functions on which VB-mDE performs better than, similarly to, or worse than MDEVM, respectively.

4.3. Friedman ranks of VB-mDE and MDEVM with different mutation strategies

To compare the performances of VB-mDE and MDEVM algorithms with five different mutation strategies, Figure 2 plots the Friedman ranks [24] of these algorithms on all the 30 functions .

As can be seen from Figure 2, for all four problem dimensions D, VB-mDE/a/b always has a smaller rank than its counterpart MDEVM/a/b.

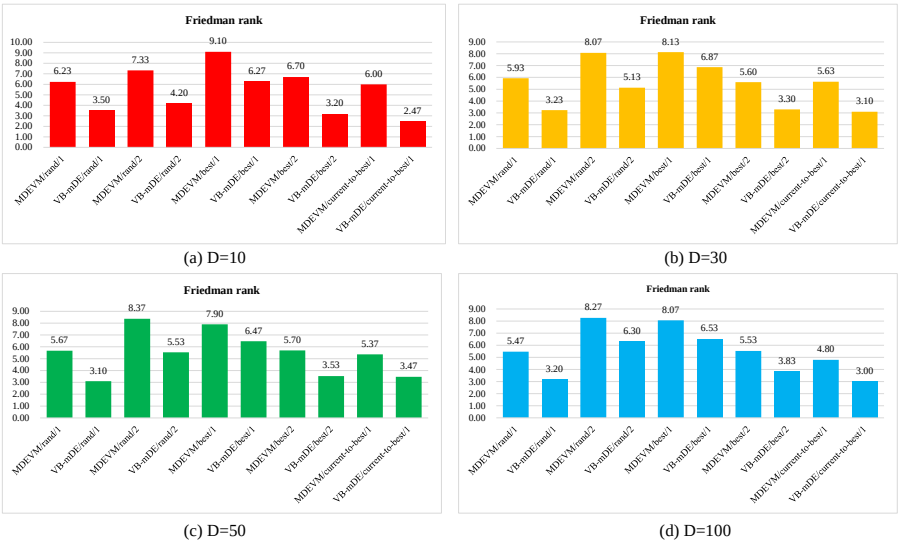
When $D = 10$, VB-mDE/current-to-best/1 achieves the best rank (i.e., 2.47), VB-mDE/best/2 the second (i.e., 3.20) , followed by VB-mDE/rand/1 (i.e., 3.50) and others.

When $D = 30$, VB-mDE/current-to-best/1 achieves the best rank (i.e., 3.10), VB-mDE/rand/1 the second (i.e., 3.23), followed by VB-mDE/best/2 (i.e., 3.30) and others.

When $D = 50$, VB-mDE/rand/1 achieves the best rank (i.e., 3.10), VB-mDE/current-to-best/1 the second (i.e., 3.47), followed by VB-mDE/best/2 (i.e., 3.53) and others.

When $D = 100$, VB-mDE/current-to-best/1 achieves the best rank (i.e., 3.00), VB-mDE/rand/1 the second (i.e., 3.20), followed by VB-mDE/best/2 (i.e., 3.83) and others.

From the analysis of Friedman test ranks, it is clear that VB-mDE/rand/1 and VB-mDE/current-to-best/1 achieve the overall best performance among the ten mDE algorithms. Thus, in the following sections, VB-mDE/rand/1 and VB-mDE/current-to-best/1 will be selected for further comparison.



(a) D=10 (b) D=30

(c) D=50 (d) D=100

Figure 2. Friedman ranks of VB-mDE and MDEVM algorithms

4.4. Convergence comparison between VB-mDE and MDEVMM

Figure 3 plots the convergence graphs of VB-mDE and MDVEM with mutation strategies rand/1 and current-to-best/1 on some typical functions with $D = 30$.

As can be seen from Figure 3, VB-mDE/a/b has relatively faster convergence speed than MDEVM/a/b on these functions.

4.5. Comparisons between VB-mDE and other mDEs

The performances of VB-mDE/rand/1 and VB-mDE/current-to-best/1 are further compared with other four mDE algorithms, i.e., DESP [9], EMDE [14], OEMDE [18] and μJADE [6]. DESP introduces the disturbance with adaptive step size to differential mutation strategy. EMDE is an improved mDE with ensemble mutation strategies. OEMDE uses both ensemble mutation strategies and opposition-based learning to enhance its performance. μJADE is a small population version of JADE algorithm with very competitive performance.

Table 4 presents the comparisons between VB-mDE and other mDEs with four different dimensions (D=10, 30, 50 and 100).

When D=10, VB-mDE/rand/1 outperforms DESP, EMDE, OEMDE and μJADE on 23, 20, 24 and 10 functions, respectively; but loses to them only on 2, 1, none and 12 functions, respectively. VB-mDE/current-to-best/1 outperforms DESP, EMDE, OEMDE and μJADE on 25, 22, 27 and 13 functions, respectively; but loses to them only on 2, 1, none and 15 functions, respectively.

When D=30, VB-mDE/rand/1 outperforms DESP, EMDE, OEMDE and μJADE on 23, 20, 25 and 12 functions, respectively; but loses to them only on 2, 2, 1 and 9 functions, respectively. VB-mDE/current-to-best/1 outperforms DESP, EMDE, OEMDE and μJADE on 25, 19, 27 and 13 functions, respectively; but loses to them only on 4, 2, 1 and 7 functions, respectively.

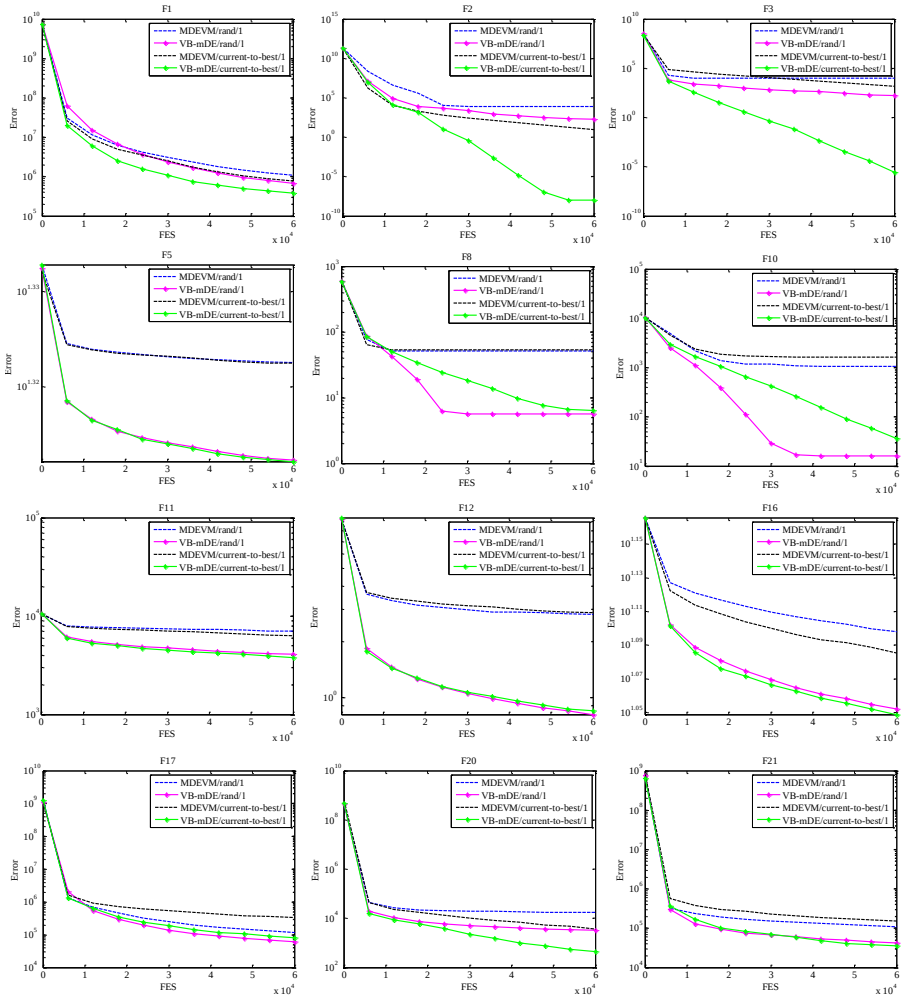


Figure 3. Convergence graphs of VB-mDE and MDEVM algorithms

When D=50, VB-mDE/rand/1 outperforms DESP, EMDE, OEMDE and μJADE on 23, 17, 25 and 17 functions, respectively; but loses to them only on 2, 3, 2 and 3 functions, respectively. VB-mDE/current-to-best/1 outperforms DESP, EMDE, OEMDE and μJADE on 23, 17, 22 and 15 functions, respectively; but loses to them only on 2, 5, 1 and 10 functions, respectively.

When D=100, VB-mDE/rand/1 outperforms DESP, EMDE, OEMDE and μJADE on 20, 19, 24 and 15 functions, respectively; but loses to them only on 5, 4, 2 and 8 functions, respectively. VB-mDE/current-to-best/1 outperforms DESP, EMDE, OEMDE and μJADE on 23, 24, 26 and 18 functions, respectively; but loses to them only on 3, 1, 2 and 9 functions, respectively.

When considering all four problem dimension (D=10, 30, 50 and 100), VB-mDE/rand/1 outperforms DESP, EMDE, OEMDE and μJADE on 89, 76, 98 and 54 func-

tions, respectively; but loses to them only on 11, 10, 5 and 32 functions, respectively. VB-mDE/current-to-best/1 outperforms DESP, EMDE, OEMDE and μJADE on 96, 82, 102 and 59 functions, respectively; but loses to them only on 11, 9, 4 and 41 functions, respectively.

In summary, based on the above analysis, it is clear that VB-mDE largely exhibits better performance than DESP, EMDE and OEMDE with four different dimensions (D=10, 30, 50 and 100). Compared with μJADE, when the dimension is small (i.e., D=10), VB-mDE is slightly worse than μJADE; when the dimension increases, VB-mDE exhibits better performance than μJADE.

Table 4. Comparisons between VB-mDE and other mDEs with different problem dimensions

	B-S-W Algorithm / Algorithm	DESP	EMDE	OEMDE	μJADE
$D = 10$	VB-mDE/rand/1	23-5-2	20-9-1	24-6-0	10-8-12
	VB-mDE/current-to-best/1	25-3-2	22-7-1	27-3-0	13-2-15
$D = 30$	VB-mDE/rand/1	23-5-2	20-8-2	25-4-1	12-9-9
	VB-mDE/current-to-best/1	25-1-4	19-9-2	27-2-1	13-10-7
$D = 50$	VB-mDE/rand/1	23-5-2	17-10-3	25-3-2	17-10-3
	VB-mDE/current-to-best/1	23-5-2	17-8-5	22-7-1	15-5-10
$D = 100$	VB-mDE/rand/1	20-5-5	19-7-4	24-4-2	15-7-8
	VB-mDE/current-to-best/1	23-4-3	24-5-1	26-2-2	18-3-9
Total B-S-W	VB-mDE/rand/1	89-20-11	76-34-10	98-17-5	54-34-32
	VB-mDE/current-to-best/1	96-13-11	82-29-9	102-14-4	59-20-41

B, S and W represent the numbers of test functions on which VB-mDE performs better than, similarly to, or worse than its competitor, respectively.

4.6. Comparisons between VB-mDE and normal DEs

The performances of VB-mDE/rand/1 and VB-mDE/current-to-best/1 are further compared with three normal DEs, namely CoBiDE [17], SaDE [25] and JADE [19]. The normal DEs are implemented with small population size, i.e., CoBiDE ($Np = 8$), SaDE ($Np = 8$), JADE ($Np = 8$), and also with normal population size, i.e., CoBiDE ($Np = 60$), SaDE ($Np = 50$) and JADE ($Np = 50$).

Table 5 presents the comparisons between VB-mDE and DEs with four different dimensions (D=10, 30, 50 and 100).

When D=10, 30, 50 and 100, VB-mDE/rand/1 and VB-mDE/current-to-best/1 perform better than CoBiDE ($Np = 8$), SaDE ($Np = 8$) and JADE ($Np = 8$). However, VB-mDE/rand/1 and VB-mDE/current-to-best/1 perform similarly to or worse than CoBiDE ($Np = 60$), SaDE ($Np = 50$) and JADE ($Np = 50$). These make sense as mDE is designed to work on small population size whereas normal DE works on normal population size.

When considering all four problem dimensions (D=10, 30, 50 and 100), VB-mDE/rand/1 outperforms CoBiDE ($Np = 8$), SaDE ($Np = 8$), JADE ($Np = 8$), CoBiDE ($Np = 60$), SaDE ($Np = 50$) and JADE ($Np = 50$) on 70, 103, 74, 53, 41 and 10 functions, respectively; but loses to them on 24, 12, 30, 49, 52 and 94 functions, respectively. Likewise, VB-mDE/current-to-best/1 outperforms CoBiDE ($Np = 8$), SaDE ($Np = 8$),

JADE ($Np = 8$), CoBiDE ($Np = 60$), SaDE ($Np = 50$) and JADE ($Np = 50$) on 74, 102, 68, 58, 45 and 10 functions, respectively; but loses to them on 20, 9, 23, 47, 52 and 91 functions, respectively.

Based on the above analysis, it can be seen that VB-mDE generally outperforms normal DEs, if the normal DEs are implemented directly with small population size. This also reinforces the motivation that mDE algorithms should be elaborately designed, rather than simply implementing normal DE algorithms with small population size.

It can be seen that VB-mDE generally loses to normal DEs with large population size. This is perceivable because DEs with large population size can maintain better diversity. At the same time, it is also worth noting that DEs with large population size cannot be used for some resource constrained optimization applications.

Table 5. Comparisons between VB-mDE and normal DEs with different problem dimensions

	B-S-W / Algorithm	CoBiDE ($Np = 8$)	SaDE ($Np = 8$)	JADE ($Np = 8$)	CoBiDE ($Np = 60$)	SaDE ($Np = 50$)	JADE ($Np = 50$)
$D = 10$	VB-mDE/rand/1	17-8-5	24-2-4	15-6-9	18-5-7	8-8-14	3-5-22
	VB-mDE/current-to-best/1	21-6-3	25-2-3	16-9-5	20-4-6	11-6-13	4-10-16
$D = 30$	VB-mDE/rand/1	21-6-3	27-1-2	17-4-9	13-5-12	11-4-15	2-4-24
	VB-mDE/current-to-best/1	23-5-2	26-3-1	21-5-4	15-4-11	11-6-13	3-3-24
$D = 50$	VB-mDE/rand/1	19-6-5	27-1-2	17-5-8	11-6-13	11-8-11	3-5-22
	VB-mDE/current-to-best/1	18-9-3	26-3-1	17-9-4	13-4-13	11-9-10	1-3-26
$D = 100$	VB-mDE/rand/1	13-6-11	25-1-4	25-1-4	11-2-17	11-7-12	2-2-26
	VB-mDE/current-to-best/1	12-6-12	25-1-4	14-6-10	10-3-17	12-2-16	2-3-25
Total B-S-W	VB-mDE/rand/1	70-26-24	103-5-12	74-16-30	53-18-49	41-27-52	10-16-94
	VB-mDE/current-to-best/1	74-26-20	102-9-9	68-29-23	58-15-47	45-23-52	10-19-91

B, S and W represent the numbers of test functions on which VB-mDE performs better than, similarly to, or worse than normal DE, respectively.

5. Conclusions

In this paper, a novel bimodal distribution based micro-DE (VB-mDE) has been developed. In our proposed VB-mDE, the vectorized bimodal distribution mechanism is employed to adjust the control parameters. Specifically, the scale factor F and cross rate CR are adjusted by bimodal Cauchy distribution, and the scale factor F is further vectorized. In this way, our proposed VB-mDE can simultaneously conciliate global exploration and local exploitation more efficiently in the mDE.

Comprehensive experiments have been carried out to compare the proposed VB-mDE with the state-of-the-art mDEs and normal DEs using the CEC2014 benchmark functions. Firstly, it is observed that our proposed VB-mDE outperforms the state-of-the-art mDEs (including MDEVM, DESP, EMDE, OEMDE and μJADE) on most functions. This demonstrates the effectiveness of vectorized bimodal distribution mechanism in VB-mDE. Secondly, our proposed VB-mDE outperforms the normal DEs (including CoBiDE, SaDE and JADE), when the normal DEs are implemented directly with small

population size. Compared with DEs with large population size, VB-mDE is defeated in the competition.

There are several aspects worth exploring in the future. Firstly, it is worthwhile to look into other parameter adjustment mechanism to further improve the performance of mDEs. Secondly, this paper only considers the parameter adaptive mechanism for mDE. However, actually, multi-strategy adaptive mechanism may be useful for mDEs as well. Finally, mDEs may be extended to other types of problems, such as constrained, dynamic, multi-objective and real-world problems.

Appendix 1

Table A1. Mean error and standard deviation of VB-mDE and MDEVM (Np=8, D=30)

	VB-mDE/rand/1	MDEVM/rand/1		VB-mDE/best/1	MDEVM/best/1		VB-mDE/cur2best/1	MDEVM/cur2best/1	
F01	6.77E+05	1.07E+06	+	2.17E+06	1.89E+06	-	3.85E+05	7.72E+05	+
	(4.67E+05)	(8.54E+05)		(4.45E+06)	(1.13E+06)		(3.05E+05)	(1.53E+06)	
F02	1.99E+02	8.09E+03	+	4.96E+07	1.68E+04	-	0.00E+00	8.80E+00	+
	(8.30E+02)	(8.83E+03)		(1.91E+08)	(2.42E+04)		(0.00E+00)	(1.39E+01)	
F03	1.59E+02	9.33E+03	+	2.12E+03	2.03E+04	+	2.74E-06	1.50E+03	+
	(4.92E+02)	(8.92E+03)		(4.13E+03)	(1.80E+04)		(8.75E-06)	(2.44E+03)	
F04	7.38E+01	7.35E+01	=	8.26E+01	9.78E+01	=	5.63E+01	4.79E+01	=
	(4.61E+01)	(4.83E+01)		(4.49E+01)	(3.56E+01)		(4.34E+01)	(3.60E+01)	
F05	2.05E+01	2.10E+01	+	2.05E+01	2.10E+01	+	2.05E+01	2.10E+01	+
	(5.74E-02)	(5.03E-02)		(6.37E-02)	(5.04E-02)		(5.13E-02)	(4.78E-02)	
F06	1.83E+01	1.30E+01	-	1.96E+01	2.66E+01	+	1.29E+01	1.48E+01	=
	(2.86E+00)	(3.31E+00)		(3.02E+00)	(3.73E+00)		(4.23E+00)	(4.20E+00)	
F07	1.86E-02	7.13E-02	+	4.36E-01	7.76E-01	=	4.62E-02	4.94E-02	=
	(5.24E-02)	(2.20E-01)		(9.91E-01)	(3.39E+00)		(6.71E-02)	(8.62E-02)	
F08	5.60E+00	5.13E+01	+	4.75E+01	1.06E+02	+	6.43E+00	5.35E+01	+
	(4.02E+00)	(1.52E+01)		(2.07E+01)	(3.23E+01)		(6.01E+00)	(1.54E+01)	
F09	1.13E+02	6.06E+01	-	9.59E+01	1.30E+02	+	9.22E+01	7.10E+01	-
	(1.21E+01)	(2.76E+01)		(2.06E+01)	(3.60E+01)		(1.38E+01)	(1.58E+01)	
F10	1.59E+01	1.06E+03	+	2.07E+02	2.46E+03	+	3.62E+01	1.62E+03	+
	(3.24E+01)	(4.29E+02)		(2.10E+02)	(5.31E+02)		(2.64E+01)	(6.26E+02)	
F11	4.09E+03	7.01E+03	+	3.76E+03	3.44E+03	-	3.80E+03	6.33E+03	+
	(3.86E+02)	(1.21E+03)		(3.99E+02)	(9.54E+02)		(3.26E+02)	(1.46E+03)	
F12	8.06E-01	2.82E+00	+	8.24E-01	2.47E+00	+	8.44E-01	2.88E+00	+
	(1.20E-01)	(3.47E-01)		(1.28E-01)	(1.02E+00)		(1.34E-01)	(3.38E-01)	
F13	4.39E-01	5.01E-01	+	5.05E-01	5.61E-01	=	4.65E-01	5.00E-01	=
	(8.12E-02)	(1.04E-01)		(1.16E-01)	(1.57E-01)		(7.42E-02)	(1.04E-01)	
F14	2.90E-01	3.49E-01	=	4.54E-01	4.30E-01	=	2.64E-01	3.48E-01	+
	(4.43E-02)	(1.58E-01)		(7.60E-01)	(2.71E-01)		(4.73E-02)	(1.51E-01)	
F15	1.16E+01	1.63E+01	=	1.31E+02	3.70E+02	+	1.73E+01	1.64E+01	=
	(1.47E+00)	(1.03E+01)		(3.55E+02)	(3.74E+02)		(6.90E+00)	(1.26E+01)	
F16	1.13E+01	1.25E+01	+	1.10E+01	1.13E+01	+	1.12E+01	1.22E+01	+
	(3.54E-01)	(7.49E-01)		(4.54E-01)	(7.85E-01)		(4.35E-01)	(5.71E-01)	
F17	6.04E+04	1.16E+05	+	2.24E+05	3.81E+05	+	7.91E+04	3.38E+05	+
	(4.10E+04)	(6.97E+04)		(4.50E+05)	(2.32E+05)		(7.19E+04)	(2.91E+05)	
F18	3.31E+03	2.68E+03	=	4.46E+06	4.08E+03	=	1.98E+03	3.65E+03	=
	(3.95E+03)	(3.24E+03)		(2.26E+07)	(4.34E+03)		(2.22E+03)	(4.28E+03)	
F19	1.02E+01	1.66E+01	=	2.74E+01	2.72E+01	-	1.18E+01	1.26E+01	=
	(1.48E+01)	(2.31E+01)		(4.72E+01)	(2.86E+01)		(1.62E+01)	(1.73E+01)	
F20	3.32E+03	1.74E+04	+	6.15E+03	3.70E+04	+	4.38E+02	3.67E+03	+
	(4.20E+03)	(1.17E+04)		(7.08E+03)	(1.99E+04)		(4.45E+02)	(3.84E+03)	
F21	4.22E+04	1.09E+05	+	8.09E+04	2.97E+05	+	3.60E+04	1.51E+05	+
	(4.53E+04)	(8.50E+04)		(1.00E+05)	(2.81E+05)		(3.74E+04)	(1.88E+05)	
F22	2.86E+02	3.30E+02	=	4.35E+02	6.17E+02	+	2.61E+02	2.72E+02	=
	(1.17E+02)	(2.18E+02)		(2.22E+02)	(2.41E+02)		(1.11E+02)	(1.15E+02)	
F23	3.15E+02	3.15E+02	+	3.16E+02	3.15E+02	=	3.15E+02	3.15E+02	+
	(6.62E-12)	(1.11E-06)		(1.01E+00)	(2.59E-01)		(2.84E-11)	(6.05E-10)	
F24	2.32E+02	2.45E+02	+	2.52E+02	2.59E+02	+	2.43E+02	2.45E+02	=
	(6.36E+00)	(6.27E+00)		(8.18E+00)	(9.50E+00)		(6.95E+00)	(8.23E+00)	
F25	2.06E+02	2.08E+02	+	2.15E+02	2.18E+02	=	2.09E+02	2.10E+02	=
	(1.87E+00)	(3.38E+00)		(5.96E+00)	(8.45E+00)		(3.89E+00)	(4.48E+00)	
F26	1.06E+02	1.10E+02	+	1.39E+02	1.54E+02	+	1.36E+02	1.22E+02	=
	(2.37E+01)	(2.99E+01)		(5.60E+01)	(6.48E+01)		(4.81E+01)	(4.14E+01)	
F27	4.98E+02	6.05E+02	+	7.73E+02	9.21E+02	+	4.92E+02	6.35E+02	+
	(1.48E+02)	(1.29E+02)		(2.16E+02)	(2.86E+02)		(1.14E+02)	(1.52E+02)	
F28	9.75E+02	1.08E+03	+	1.35E+03	1.99E+03	+	9.84E+02	1.09E+03	+
	(5.66E+01)	(1.65E+02)		(3.15E+02)	(5.52E+02)		(9.17E+01)	(1.19E+02)	
F29	1.36E+03	1.28E+03	=	2.01E+03	1.57E+03	=	1.17E+03	1.39E+03	+
	(3.71E+02)	(3.27E+02)		(2.66E+03)	(6.81E+02)		(3.88E+02)	(4.57E+02)	
F30	2.13E+03	2.77E+03	+	3.62E+03	5.04E+03	+	2.49E+03	2.44E+03	=
	(7.60E+02)	(9.04E+02)		(9.64E+02)	(3.77E+03)		(8.97E+02)	(1.07E+03)	
B-S-W		21-7-2			18-8-4			17-12-1	

+, = and - symbolize VB-mDE being better than, similar to, or worse than MDEVM, respectively.

Table A2. Mean error and standard deviation of VB-mDE and other mDEs (Np=8, D=30)

	VB-mDE/rand/1	DESP		EMDE		OEMDE		μJADE	
F01	6.77E+05	3.20E+06	+	7.12E+05	=	6.61E+07	+	1.08E+07	+
	(4.67E+05)	(1.57E+06)		(4.98E+05)		(2.68E+08)		(6.24E+06)	
F02	1.99E+02	1.09E+04	+	8.35E+03	+	1.06E+04	+	4.91E-10	-
	(8.30E+02)	(8.84E+03)		(7.99E+03)		(9.91E+03)		(3.51E-09)	
F03	1.59E+02	7.10E+04	+	8.66E+03	+	6.02E+05	+	6.50E-04	-
	(4.92E+02)	(1.89E+04)		(8.20E+03)		(3.54E+06)		(2.63E-03)	
F04	7.38E+01	1.25E+02	+	6.08E+01	=	8.34E+01	=	5.88E+01	-
	(4.61E+01)	(3.82E+01)		(3.45E+01)		(4.14E+01)		(4.48E+01)	
F05	2.05E+01	2.03E+01	-	2.10E+01	+	2.10E+01	+	2.04E+01	-
	(5.74E-02)	(4.17E-01)		(5.18E-02)		(2.02E-01)		(1.84E-01)	
F06	1.83E+01	2.44E+01	+	1.61E+01	-	2.25E+01	=	2.62E+01	+
	(2.86E+00)	(3.36E+00)		(3.48E+00)		(9.89E+00)		(1.71E+00)	
F07	1.86E-02	1.27E-02	=	2.23E-01	+	1.00E-01	+	6.03E-03	=
	(5.24E-02)	(1.59E-02)		(1.16E+00)		(1.65E-01)		(1.19E-02)	
F08	5.60E+00	1.07E+02	+	6.36E+01	+	6.22E+01	+	3.31E+01	+
	(4.02E+00)	(2.43E+01)		(1.62E+01)		(1.87E+01)		(8.87E+00)	
F09	1.13E+02	1.34E+02	+	7.66E+01	-	8.54E+01	-	1.62E+02	+
	(1.21E+01)	(3.52E+01)		(2.52E+01)		(2.48E+01)		(1.87E+01)	
F10	1.59E+01	2.67E+03	+	1.38E+03	+	3.56E+03	+	1.05E+03	+
	(3.24E+01)	(7.16E+02)		(4.91E+02)		(2.06E+03)		(3.79E+02)	
F11	4.09E+03	3.38E+03	-	5.84E+03	+	6.48E+03	+	5.32E+03	+
	(3.86E+02)	(8.16E+02)		(2.23E+03)		(1.55E+03)		(7.49E+02)	
F12	8.06E-01	1.05E+00	+	2.83E+00	+	2.15E+00	+	8.66E-01	=
	(1.20E-01)	(4.36E-01)		(3.49E-01)		(1.23E+00)		(5.46E-01)	
F13	4.39E-01	4.28E-01	=	4.90E-01	+	5.25E-01	+	4.57E-01	=
	(8.12E-02)	(9.84E-02)		(9.74E-02)		(1.11E-01)		(6.62E-02)	
F14	2.90E-01	2.88E-01	=	3.58E-01	=	3.49E-01	+	3.40E-01	=
	(4.43E-02)	(7.50E-02)		(1.70E-01)		(1.35E-01)		(1.57E-01)	
F15	1.16E+01	4.23E+01	+	1.75E+01	+	2.26E+01	+	1.50E+01	+
	(1.47E+00)	(2.15E+01)		(1.18E+01)		(2.77E+01)		(2.99E+00)	
F16	1.13E+01	1.14E+01	=	1.23E+01	+	1.27E+01	+	1.14E+01	=
	(3.54E-01)	(7.11E-01)		(7.57E-01)		(6.80E-01)		(9.77E-01)	
F17	6.04E+04	7.97E+05	+	8.74E+04	+	2.60E+05	+	5.10E+05	+
	(4.10E+04)	(5.26E+05)		(5.24E+04)		(6.38E+05)		(4.62E+05)	
F18	3.31E+03	3.67E+03	=	2.85E+03	=	3.70E+03	=	1.78E+03	=
	(3.95E+03)	(4.25E+03)		(3.41E+03)		(3.68E+03)		(2.79E+03)	
F19	1.02E+01	4.62E+01	+	1.59E+01	=	2.10E+01	+	8.59E+00	-
	(1.48E+01)	(4.40E+01)		(2.24E+01)		(2.43E+01)		(1.19E+00)	
F20	3.32E+03	4.89E+04	+	1.11E+04	+	1.46E+04	+	2.70E+02	-
	(4.20E+03)	(2.72E+04)		(8.53E+03)		(1.01E+04)		(2.70E+02)	
F21	4.22E+04	2.86E+05	+	4.91E+04	=	1.09E+05	+	2.31E+04	-
	(4.53E+04)	(2.28E+05)		(3.82E+04)		(7.21E+04)		(1.61E+04)	
F22	2.86E+02	5.92E+02	+	3.77E+02	=	4.19E+02	+	2.94E+02	=
	(1.17E+02)	(2.17E+02)		(2.15E+02)		(2.10E+02)		(1.43E+02)	
F23	3.15E+02	3.20E+02	+	3.15E+02	+	3.15E+02	+	3.15E+02	-
	(6.62E-12)	(4.14E+00)		(4.33E-10)		(4.47E-05)		(3.95E-13)	
F24	2.32E+02	2.61E+02	+	2.47E+02	+	2.46E+02	+	2.28E+02	-
	(6.36E+00)	(1.03E+01)		(7.45E+00)		(7.32E+00)		(3.98E+00)	
F25	2.06E+02	2.24E+02	+	2.10E+02	+	3.03E+02	+	2.11E+02	+
	(1.87E+00)	(6.15E+00)		(5.01E+00)		(8.39E+01)		(2.52E+00)	
F26	1.06E+02	1.63E+02	+	1.32E+02	+	2.56E+02	+	1.08E+02	=
	(2.37E+01)	(4.88E+01)		(4.63E+01)		(1.16E+02)		(2.74E+01)	
F27	4.98E+02	7.51E+02	+	7.20E+02	+	7.32E+02	+	5.43E+02	+
	(1.48E+02)	(2.79E+02)		(1.33E+02)		(2.92E+02)		(1.74E+02)	
F28	9.75E+02	2.40E+03	+	1.23E+03	+	2.15E+03	+	1.03E+03	=
	(5.66E+01)	(5.58E+02)		(2.42E+02)		(1.78E+03)		(1.35E+02)	
F29	1.36E+03	1.39E+07	+	1.36E+03	=	1.23E+07	=	6.68E+05	+
	(3.71E+02)	(2.55E+07)		(3.91E+02)		(8.79E+07)		(2.30E+06)	
F30	2.13E+03	2.35E+04	+	2.82E+03	+	3.13E+03	+	3.06E+03	+
	(7.60E+02)	(1.27E+04)		(8.58E+02)		(8.88E+02)		(9.45E+02)	
B-S-W		23-5-2		20-8-2		25-4-1		12-9-9	

+, = and - symbolize VB-mDE being better than, similar to, or worse than its competitor, respectively.

References

[1] S. Das, S.S. Mullick and P.N. Suganthan, Recent advances in differential evolutionan updated survey, *Swarm and Evolutionary Computation* **27** (2016), 1-30.

[2] X. Chen, W. Du and F. Qian, Solving chemical dynamic optimization problems with ranking-based differential evolution algorithms, *Chinese Journal of Chemical Engineering* **24**(11) (2016), 1600-1608.

[3] D. Zou, S. Li, G.G. Wang, Z. Li and H. Ouyang, An improved differential evolution algorithm for the economic load dispatch problems with or without valve-point effects, *Applied Energy* **181** (2016), 375-390.

[4] Q.K. Pan, L. Wang, L. Gao and W.D. Li, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, *Information Sciences* **181**(3)(2011), 668-685.

[5] J. Ronkkonen, S. Kukkonen and K.V. Price, Real-parameter optimization with differential evolution, in: *2005 IEEE congress on evolutionary computation*, IEEE, 2005, pp.506-513.

[6] C. Brown, Y. Jin, M. Leach and M. Hodgson, μ JADE: adaptive differential evolution with a small population, *Soft computing* **20**(10)(2016), 4111-4120.

[7] H. Salehinejad and S. Talebi, Dynamic fuzzy logic-ant colony system-based route selection system, *Applied Computational Intelligence and Soft Computing*, (2010), Article ID 428270, 13 pages.

[8] S. Rahnamayan and H.R. Tizhoosh. Image thresholding using micro opposition-based differential evolution (micro-ODE), in: *2008 IEEE Congress on Evolutionary Computation*, IEEE, 2008, pp.1409-1416.

[9] X. Ren, Z. Chen and Z. Ma, Differential evolution using smaller population, in: *2010 Second International Conference on Machine Learning and Computing*, IEEE, 2010, pp. 76-80.

[10] I. Fajfar, T. Tuma, J. Puhan, J. Olensek and A. Burmen, Towards smaller populations in differential evolution, *J Microelectron Electron Compon Mater* **42** (2012), 152-163.

[11] M. Olguin-Carbajal, E. Alba and J. Arellano-Verdejo, Micro-differential evolution with local search for high dimensional problems, in: *2013 IEEE Congress on Evolutionary Computation*, IEEE, 2013, pp. 48-54.

[12] Y.E. Yildiz and A.O. Topal, Large scale continuous global optimization based on micro differential evolution with local directional search, *Information Sciences* **477** (2019),533-544.

[13] H. Salehinejad, S. Rahnamayan and H.R. Tizhoosh, Micro-differential evolution: Diversity enhancement and a comparative study, *Applied Soft Computing* **52** (2017), 812-833.

[14] H. Salehinejad, S. Rahnamayan and H.R. Tizhoosh, Exploration enhancement in ensemble micro-differential evolution, in: *IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2016, pp. 63-70.

[15] R.D. Al-Dabbagh, F. Neri, N. Idris and M.S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, *Swarm and Evolutionary Computation* **43** (2018), 284-311.

[16] R. Tanabe and A. Fukunaga, Reviewing and benchmarking parameter control methods in differential evolution, *IEEE Transactions on Cybernetics* **50**(3)(2019),1170-1184.

[17] Y. Wang, H.X. Li, T. Huang and L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, *Applied Soft Computing* **18** (2014), 232-247.

[18] H. Salehinejad, S. Rahnamayan and H.R. Tizhoosh, Opposition-based ensemble micro-differential evolution, in: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2017, pp.1-8.

[19] J. Zhang and A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation* **13**(5) (2009), 945-958.

[20] Y.L. Li, Z.H. Zhan, Y.J. Gong and J. Zhang, Fast micro-differential evolution for topological active net optimization, *IEEE Transactions on Cybernetics* **46** (6) (2015), 1411-1423.

[21] E. Mininno, F. Neri, F. Cupertino and D. Naso, Compact differential evolution, *IEEE Transactions on Evolutionary Computation* **15** (1) (2010), 32-54.

[22] J. Brest, S. Greiner, B. Boskovic and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation* **10** (6) (2006), 646-657.

[23] J.J. Liang, B.Y. Qu and P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, *Computational Intelligence Laboratory, Zhengzhou University, China and Technical Report, Nanyang Technological University, Singapore* (2013), 635.

[24] J. Alcal-Fdez, L. Snchez, S. Garca, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernndez and F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, *Soft Computing* **13** (3) (2009), 307-318.

[25] A.K. Qin, V.L. Huang and P.N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE transactions on Evolutionary Computation* **13**(2) (2008), 398-417.

Authors' Bios

Xu Chen is an Associate Professor in School of Electrical and Information Engineering at Jiangsu University, China. He received the Ph.D. degree in Control Science and Engineering from East China University of Science and Technology, China. His research interests focus on evolutionary computation techniques and power system optimization.

Xueliang Miao received the bachelor's degree in Electrical Engineering & Automation from Jiangsu University in 2017, and received the master's degree in Control Engineering from Jiangsu University in 2020. He is now an engineer of China nuclear power technology corporation, LTD. His research interests focus on intelligent optimization algorithms and nondestructive testing.

Hugo Tianfield is a Professor of Computing at Glasgow Caledonian University, Glasgow, United Kingdom from March 2001. Prof Tianfield is extensively involved in professional activities. He is member of EPSRC Peer College UK, Chair of the Technical Committee on Cyber-Physical Cloud Systems with the IEEE Systems, Man, and

Cybernetics Society, Editor-in-Chief of Multiagent and Grid Systems -An International Journal, and Associate Editor of IEEE Transactions on Systems, Man, and Cybernetics-Systems. Prof Tianfield is Director of AI Research Lab. His research areas include Big Data, Cloud Computing, Cyber Security, and Internet of Things. He is (co-)author of about 200 research publications in refereed journals and conferences, and is a frequent invited speaker at international forums and institutions worldwide.